

FIG. 1

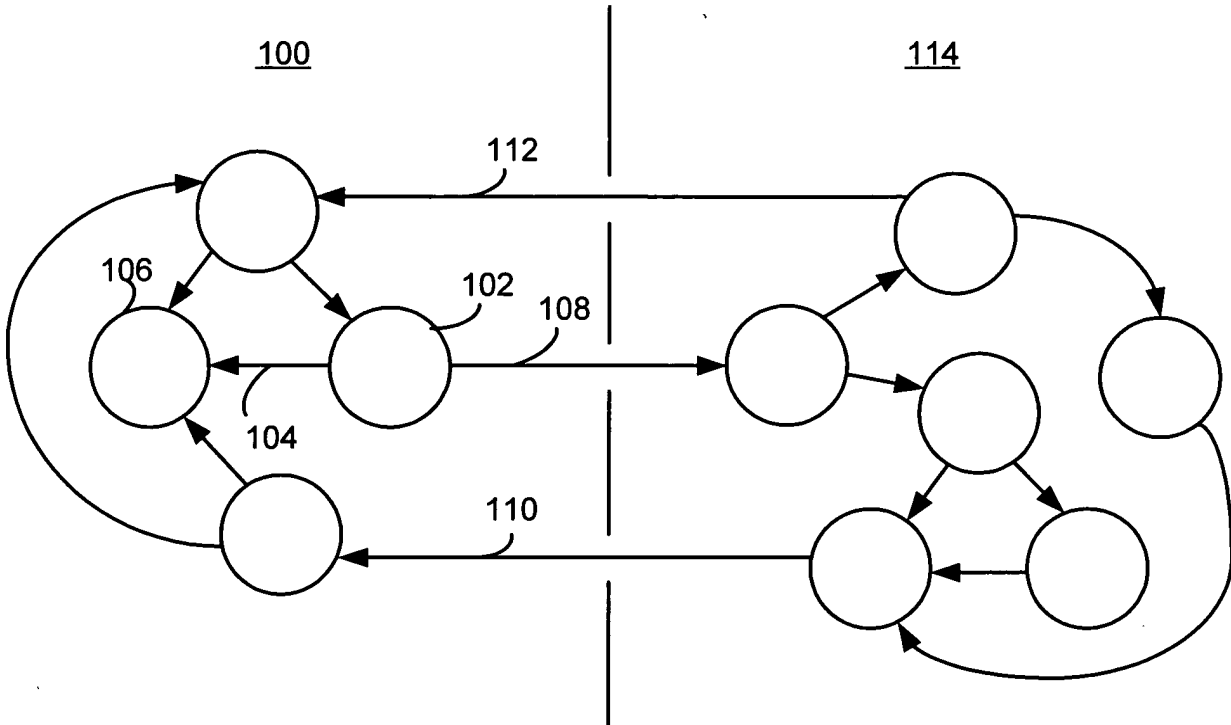


FIG. 2

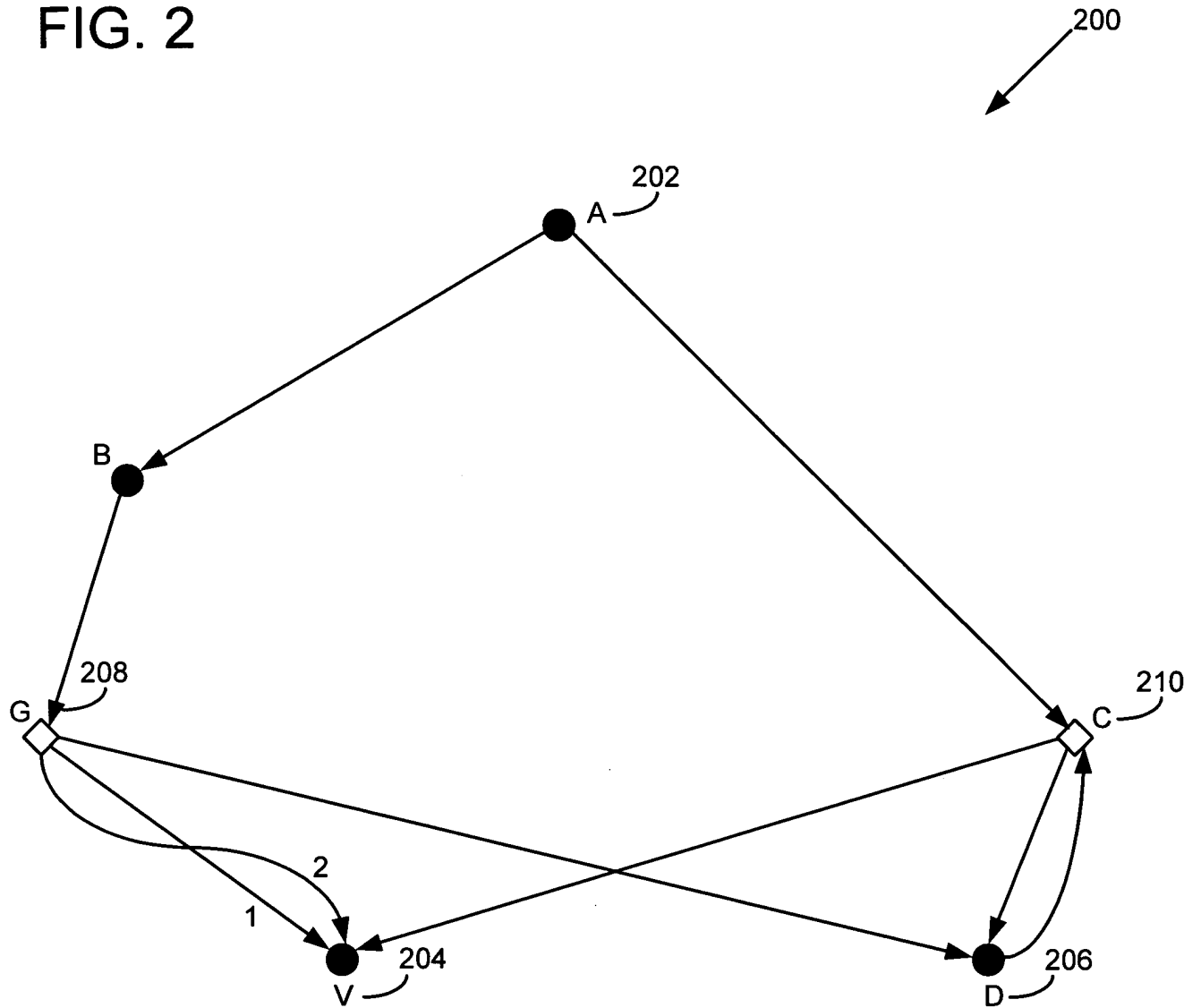


FIG. 3

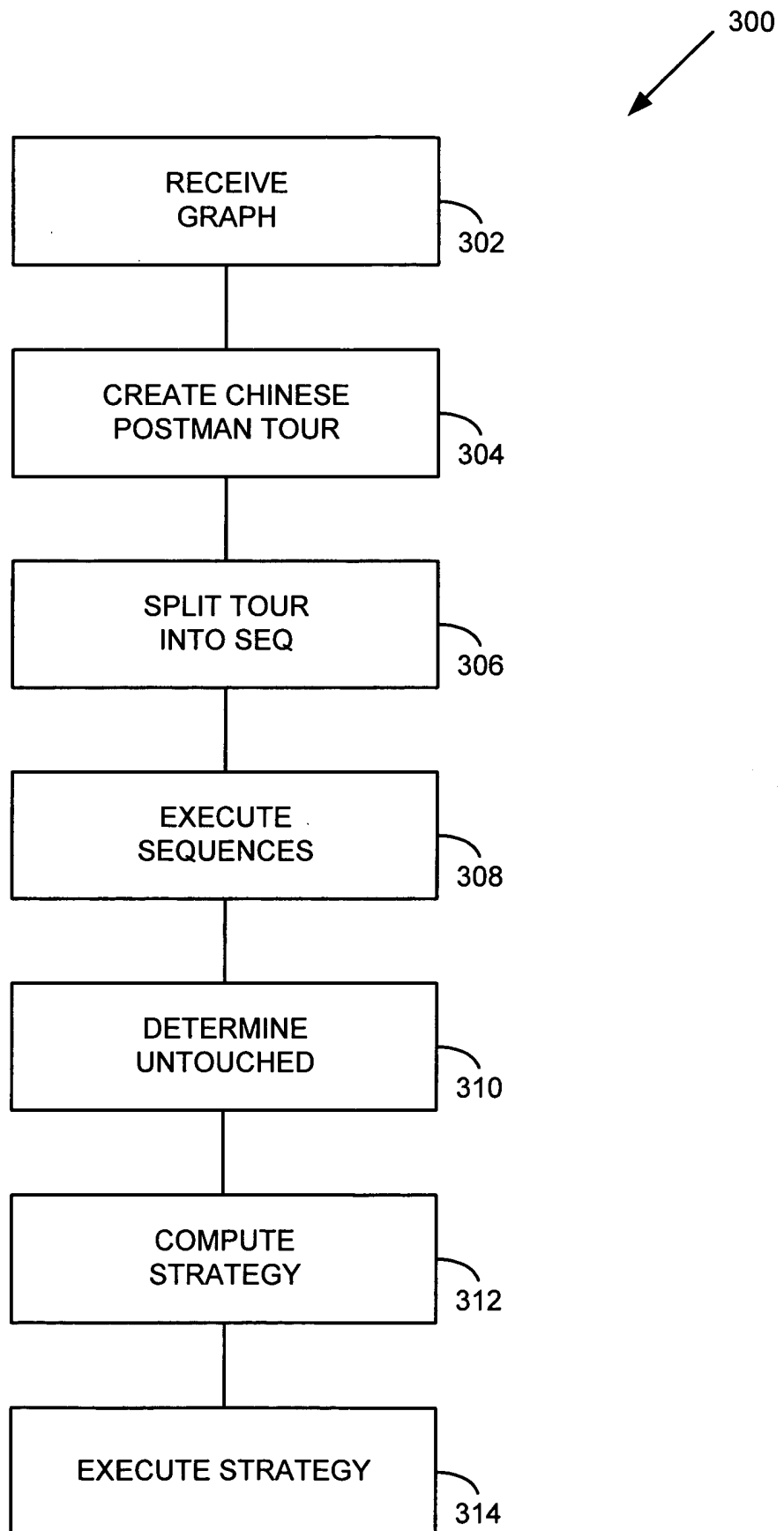


FIG. 4

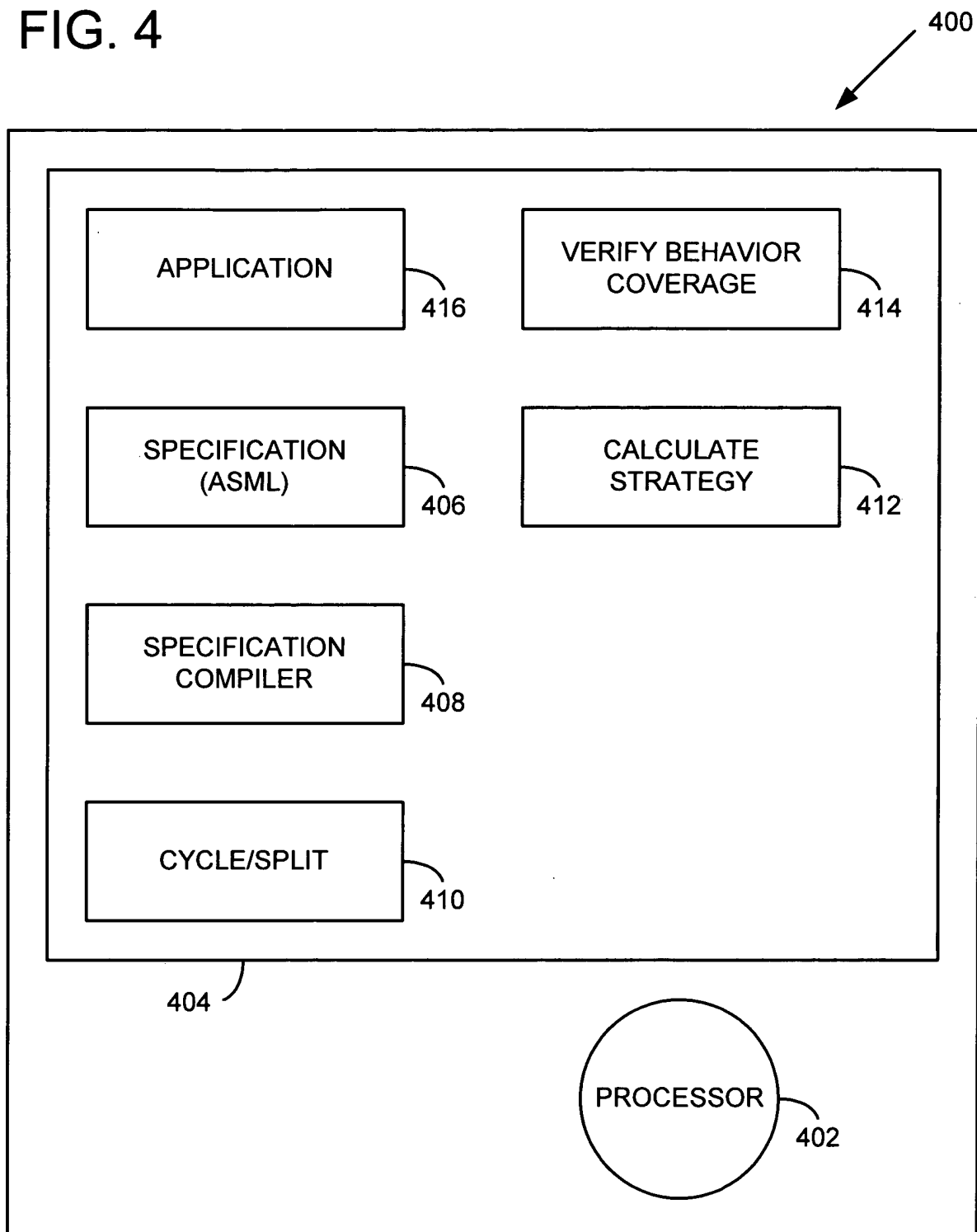


FIG. 5

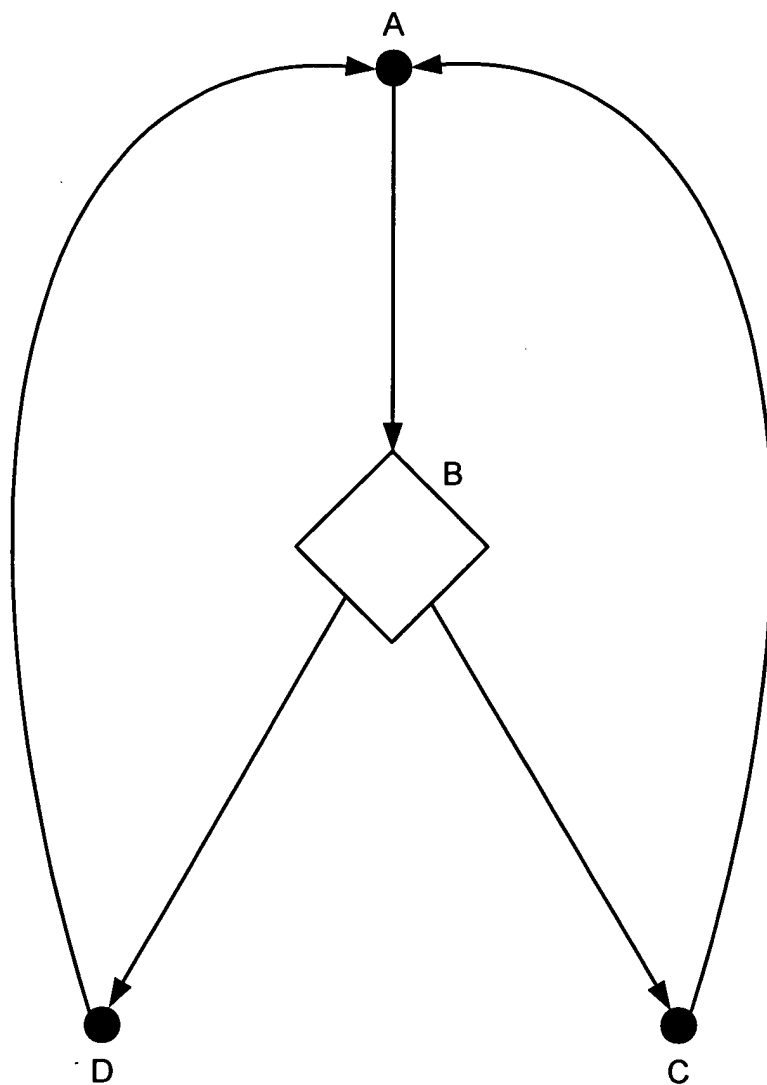


FIG. 6

600



```
Initialize() {  
602   front:=P // P is victory set  
604   newfront:={}  
606  
608   foreach v in P // initialize vertices of victory set  
610     foreach i from 0 to n { // n is number of maximum edges allowed  
612       Pr(v,i):=1 // vertices of victory set have probability of 1  
614       C(v,i):=0 // no edge costs since this is victory set  
616       S(v,i):=null // no strategies leave the victory set  
618     }  
  
620   foreach v in V-P {  
622     for each i from 0 to n {  
624       Pr(v,i):=0 // all other vertices initialized with zero probability,  
626       C(v,i):=0 //                               cost and strategy  
628       S(v,i):=null //  
    }  
  }  
}
```

FIG. 7

VERTEX	PROB COST STRATEGY	N = 0... 4				
		0	1	2	3	4
V	PR	1	1	1	1	1
	C	0	0	0	0	0
	S	NULL	NULL	NULL	NULL	NULL
A	PR	0	0	0	0	0
	C	0	0	0	0	0
	S	NULL	NULL	NULL	NULL	NULL
B	PR	0	0	0	0	0
	C	0	0	0	0	0
	S	NULL	NULL	NULL	NULL	NULL
G	PR	0	0	0	0	0
	C	0	0	0	0	0
	S	NULL	NULL	NULL	NULL	NULL
C	PR	0	0	0	0	0
	C	0	0	0	0	0
	S	NULL	NULL	NULL	NULL	NULL
D	PR	0	0	0	0	0
	C	0	0	0	0	0
	S	NULL	NULL	NULL	NULL	NULL

FIG. 8

800



```
802 StrategyCalculation(n){
804   for (i=1; i<=n; i++){
806     foreach v in front
808       Process(v,i)
810     foreach v in newfront
812       foreach k:=i+1;k<=n;k++ {
814         S(v,k):=S(v,i)
816         Pr(v,k):=Pr(v,i)
818         C(v,k):=C(v,i)
           }

820   Visited+=newfront
822   front:=newfront
824   newfront:={}

      }
    }
```


FIG. 9

900



```
902 Process(vertex v,i){
904   foreach edge entering v {
906     let u=edge.source // u is the source vertex of edge
908     if u is not in P // never exit from P, the victory set
910       if u is deterministic
912         if ImprovingOnEdge( edge,i){
914           S(u,i):=edge
916           Pr(u,i):=Pr(v,i-1)
918           C(u,i):=cost(edge)+C(v, i-1)
920           newfront:=newfront U {u} // add u to newfront
          }
922       else {//the node edge.source is nondeterministic


924         oldPr=Pr(u,i)
926         oldC= C(u,i)
928         if i>1 then
930           Pr(u,i)+=p(edge)(Pr(v,i-1)-Pr(v,i-2))
          else
932           Pr(u,i)+=p(edge)Pr(v,i-1)

934           C(u,i) = max{cost(e)+ C(e.target,i-1): e exits from u}

936           if( oldPr ≠ Pr(u,i) or oldC ≠ C(u,i))
938             newfront:=newfront U {u}
          }
        }
      }
    }

940 bool ImprovingOnEdge(edge, v, i)
    return
942 (Pr(edge.target,i-1),cost(edge)+ C(edge.target,i-1)) <
    (Pr(edge.source,i),C(edge.source,i))
```

FIG. 10

1000


VERTEX	PROB COST STRATEGY	N = 0... 4				
		0	1	2	3	4
V	PR	1	1	1	1	1
	C	0	0	0	0	0
	S	NULL	NULL	NULL	NULL	NULL
A	PR	0	0	1/2	2/3	3/4
	C	0	0	2	3	4
	S	NULL	NULL	AC	AB	AC
B	PR	0	0	2/3	2/3	2/3
	C	0	0	2	2	2
	S	NULL	NULL	BG	BG	BG
G	PR	0	2/3	2/3	2/3	2/3
	C	0	1	1	1	1
	S	NULL	NULL	NULL	NULL	NULL
C	PR	0	1/2	1/2	1/2	1/2
	C	0	1	1	1	1
	S	NULL	NULL	NULL	NULL	NULL
D	PR	0	0	1/2	1/2	1/2
	C	0	0	2	2	2
	S	NULL	NULL	DC	NULL	NULL

FIG. 11

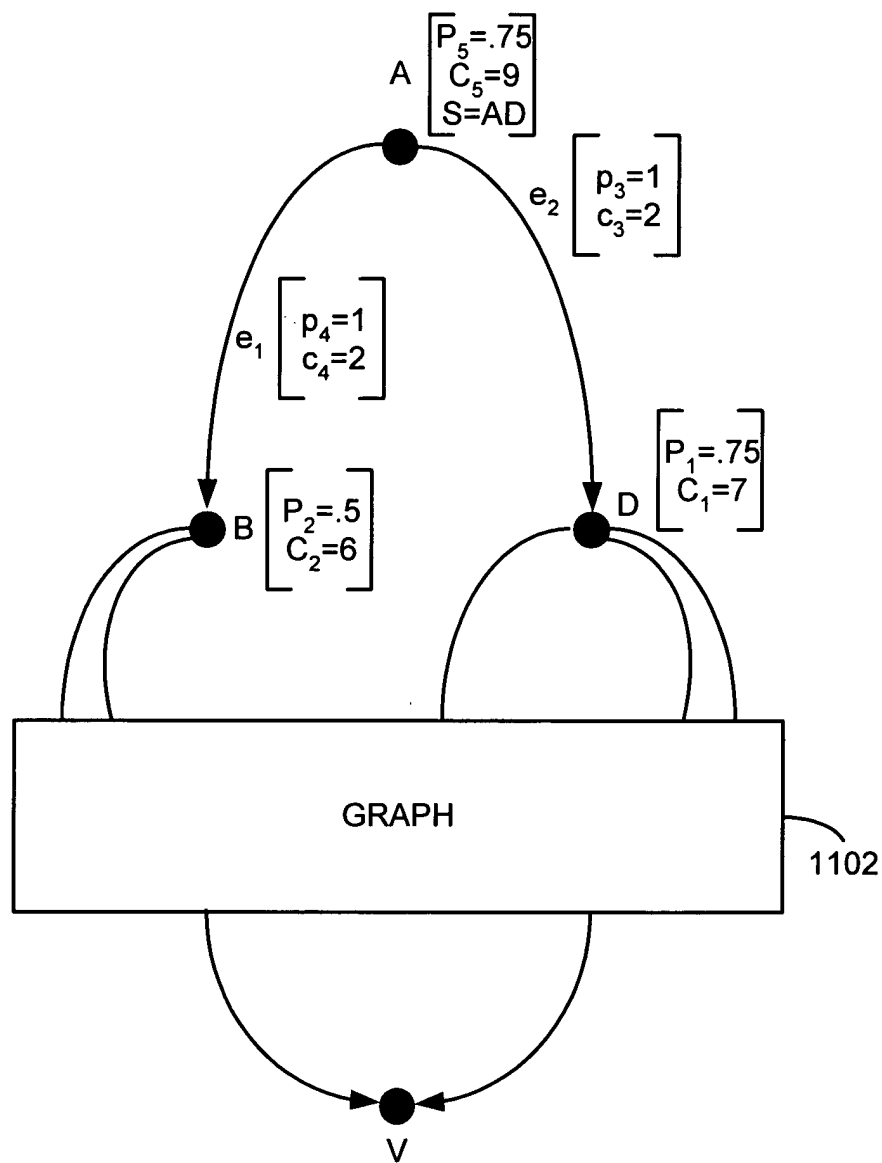


FIG. 12

```

    Traverse(node v, integer k){
        while (k>0){
            k:=k-1
            let e=edge choosen by the environment

            Ti:=choose any Ti starting with e
            cover every edge of Ti
            v:=end of Ti
        }
    }

```

FIG. 13

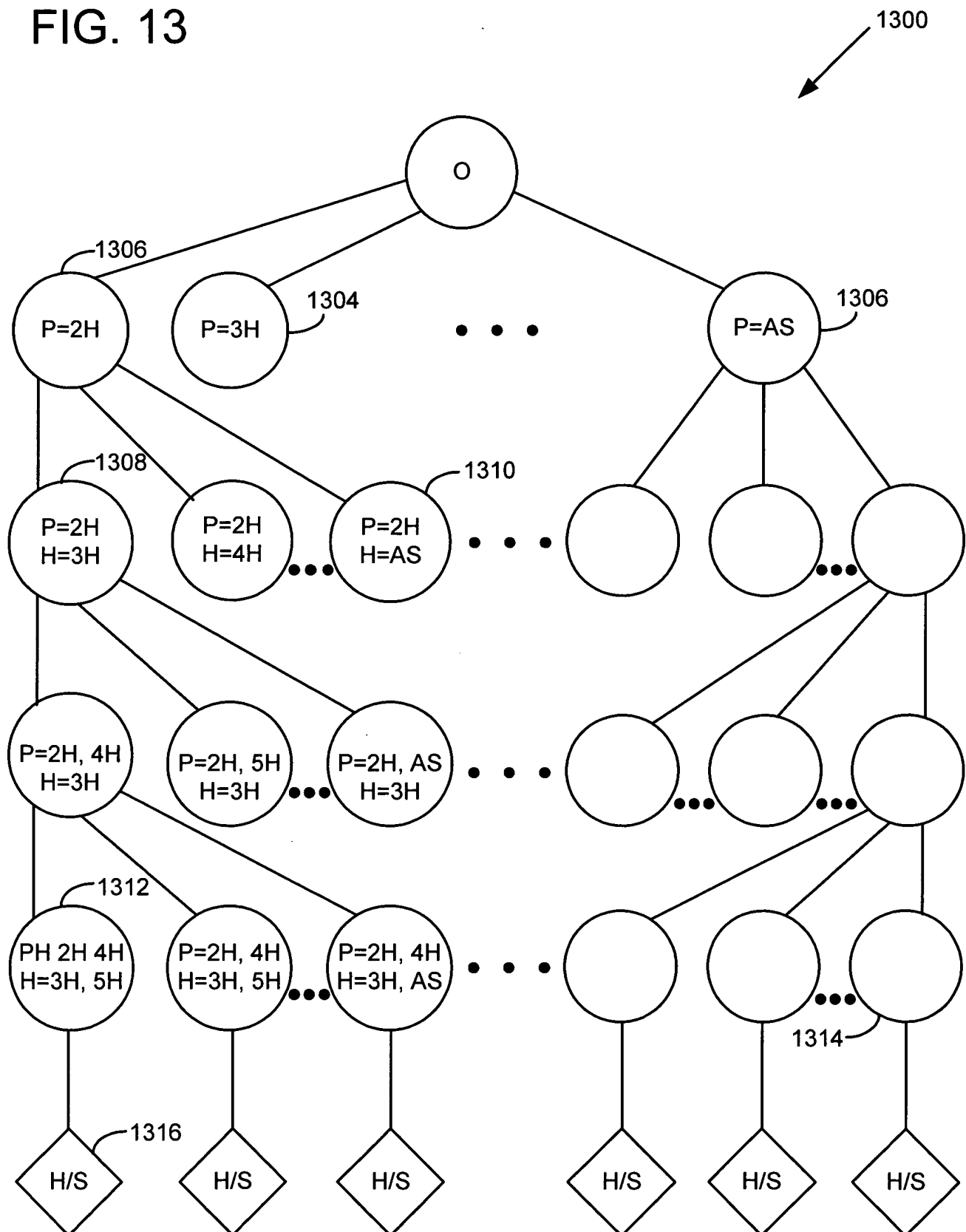


FIG. 14

